SoK
ATT&CK techniques and trends in windows malware

Oosthoek, Kris; Doerr, Christian

DOI
10.1007/978-3-030-37228-6_20

Publication date
2019

Document Version
Final published version

Published in
Security and Privacy in Communication Networks - 15th EAI International Conference, SecureComm 2019, Proceedings

Citation (APA)
Oosthoek, K., & Doerr, C. (2019). SoK: ATT&CK techniques and trends in windows malware. In S. Chen, K.-K. R. Choo, X. Fu, W. Lou, & A. Mohaisen (Eds.), *Security and Privacy in Communication Networks - 15th EAI International Conference, SecureComm 2019, Proceedings* (Vol. 304, pp. 406-425). (Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST; Vol. 304 LNICST). Springer. https://doi.org/10.1007/978-3-030-37228-6_20

SoK: ATT&CK Techniques and Trends in Windows Malware

Kris Oosthoek(✉) and Christian Doerr

Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
{k.oosthoek,c.doerr}@tudelft.nl
http://www.cyber-threat-intelligence.com

Abstract. In an ever-changing landscape of adversary tactics, techniques and procedures (TTPs), malware remains the tool of choice for attackers to gain a foothold on target systems. The Mitre ATT&CK framework is a taxonomy of adversary TTPs. It is meant to advance cyber threat intelligence (CTI) by establishing a generic vocabulary to describe post-compromise adversary behavior. This paper discusses the results of automated analysis of a sample of 951 Windows malware families, which have been plotted on the ATT&CK framework. Based on the framework's tactics and techniques we provide an overview of established techniques within Windows malware and techniques which have seen increased adoption over recent years. Within our dataset we have observed an increase in techniques applied for fileless execution of malware, discovery of security software and DLL side-loading for defense evasion. We also show how a sophisticated technique, command and control (C&C) over IPC named pipes, is getting adopted by less sophisticated actor groups. Through these observations we have identified how malware authors are innovating techniques in order to bypass established defenses.

Keywords: Malware analysis · ATT&CK framework · Classification · Cyber threat intelligence · Advanced persistent threats

1 Introduction

Malware continues to spread increasingly and with serious consequences for organizations and private individuals. According to the 2018 Verizon Data Breach Investigations Report, malware is the primary attack tactic in 30% of data breaches [30]. Attackers keep innovating their TTPs to circumvent established defenses that could impede their modus operandi. As attackers continue to increase the sophistication of their techniques, the collection of CTI on attacker innovation is fundamental to inform adequate mitigation.

Obtaining such insight from malware analysis has become increasingly challenging as a result of evasion techniques such as polymorphism and metamorphism now being widely applied [3] and even available 'as a service' to cybercriminals [25]. Crimeware toolkits like Zeus have provided cybercriminals with

S. Chen et al. (Eds.): SecureComm 2019, LNICST 304, pp. 406–425, 2019.
https://doi.org/10.1007/978-3-030-37228-6_20

effective malware kits difficult to detect using conventional mitigations [4]. The vast flood of new malware embedding improved TTPs has resulted in a 'weapons race' [20] between developers of malware and anti-malware software. To obtain objective and reliable CTI on advances in malware, the study of innovation within individual malware families needs to be supplemented with an overview of innovation within the malware ecosystem by and large.

To build a common understanding of the TTPs observed through such analysis, malware research benefits from the adoption of a common taxonomy that facilitates the dissemination of CTI from malware analysis. To date the field of malware research has not reached consensus on the adoption of such a reporting standard.

For this paper, an automated analysis of samples from 951 unique families of Windows malware was performed. To discuss the results of that analysis we have mapped them onto the industry-standard ATT&CK framework by Mitre, further referred to as the ATT&CK framework. The framework describes malware techniques in their tactical context and allows for a common understanding of post-compromise malware behavior. Because it considers techniques in the context of the attack life cycle instead of viewing them as separate artifacts, use of the framework informs more effective detection and mitigation of identified techniques. This is the first study to apply an industry-accepted taxonomy to the analysis of a large corpus of malware. In doing so it provides insight in the adoption of innovative techniques for execution, discovery and C&C. We make the following contributions:

- We have applied an industry-standard CTI framework to malware analysis results in order to advance dissemination of CTI from malware analysis.
- We have studied behavior in a sample of Windows malware, through which we are able to observe how malware authors are implementing techniques.
- We have identified trends in the implementation of fileless execution of malware, discovery of security software and DLL side-loading.
- We also show how a sophisticated technique, C&C via IPC named pipes, is adopted by less sophisticated actor groups.
- We have evaluated the potential and limitations for using automated malware analysis as a source for CTI.

The remainder of this paper is structured as follows: Sect. 2 provides an overview of related work on automated analysis and malware classification. Section 3 outlines the ATT&CK framework. Section 4 describes the methodology of our analysis. Section 5 presents the results using the common language offered by ATT&CK. Section 6 presents a sophisticated technique of which we have observed increasing adoption. Section 7 describes the limitations of using automated analysis for CTI. Section 8 summarizes of our findings.

2 Related Work

Two types of previous work are important to position our work. First, other studies on automated malware analysis and second with regards to standardized reporting of malware behavior.

Automated Analysis. Among the earliest to analyze a large corpus of Windows malware were Willems et al., introducing CWSandbox and presenting its results based on the analysis of 6,148 malware binaries [31]. Bayer et al. have published about the now discontinued Anibus platform, which they used to analyze malware samples gathered in the wild from 2007 until 2008 [3]. By classifying the observed malicious behavior into common areas of host activity, they did provide important insight into the behavior of malware on a host after infection. Another significant contribution was made by Song et al., who have combined static and dynamic methods in BitBlaze, which is now also discontinued [24]. Other researchers have focused on the automated analysis of specific categories of malware such as ransomware [10], as well as evasive malware [11], encryption and packers [14] and anti-virtualization and anti-debugging in malware [5]. More recently, Grill et al. [7] conducted a study of bootkit technology based on the analysis of 29 bootkit families observed from 2006 until 2014. However, the malware landscape is subject to rapid evolution and most research studying a larger corpus of Windows malware does not include the analysis of trends.

Taxonomies. The use of a common language, also referred to as classification or taxonomy, is vital for malware analysis to inform CTI unambiguously. Researchers from academia and industry have recognized this gap. One of the earliest steps were taken by Kirilov et al. at Mitre [12], who recognized that communication of malware analysis results is impeded by the absence of a standard for the characterization of malware. They proposed the Malware Attribute Enumeration and Characterization (MAEC), which encodes malware behavior and has gained some industry adoption. Other researchers have pointed out the need for a higher-level taxonomy that explains the behavior of malware within the broader context of an attack [19]. Building on existing models from traditional areas of defense, Lockheed Martin Corporation developed the Cyber Kill Chain [8]. The philosophy behind this model is that effective mitigation of malicious activity is driven by knowledge about an actor's TTPs. The Cyber Kill Chain describes malicious activity based on seven discrete phases, but it is argued that it is insufficient in doing so. The model is criticized for emphasizing pre-intrusion activity [18] at the expense of post-intrusion activity [13, 18]. As pre-compromise activity takes place outside the network perimeter it can not be observed, as description of such activity is mostly based on assumptions. As the model does not include common attack tactics as privilege escalation and lateral movement, the Cyber Kill Chain is less suited towards the full reporting of an attack's life cycle. Researchers at Mitre recognized this gap and have published a behavioral model named ATT&CK [26], which will be discussed in the next section.

Several authors have made significant contributions studying malware using automated analysis tools. This needs to be supplemented with analysis reported using an established taxonomy to further extend the common understanding of malware behavior. This study is the first to use the industry-standard ATT&CK framework to present the results of a large representative sample of malware.

3 The Mitre ATT&CK Framework

The ATT&CK framework was originally created by Mitre as it was recognized that existing reference models were insufficient in categorizing post-compromise activity into attacker TTPs [26]. ATT&CK distinguishes between tactics and techniques. Tactics describe adversary goals, techniques are the technical means through which goals are achieved. The authors position ATT&CK as an adversary model which standardizes both the tactics and the technical capabilities used after an initial foothold has been gained. The framework design also separates pre-compromise and post-compromise activity. The PRE-ATT&CK model focuses on attacker activity prior to delivery and exploitation and is independent of any technology platform as it offers a categorization of an attacker's reconnaissance activity, most of which cannot or only partially be detected. The Enterprise model covers on techniques implemented over the full attack life cycle on Windows, Linux and macOS, platforms commonly implemented within an enterprise context. Attack techniques targeting Android and iOS are contained in a separate model. Over the last couple of years, the framework has become the industry standard for describing malware techniques and attacker campaigns. The model has been open-sourced by Mitre and is being implemented in industry products. The Mitre ATT&CK website contains a knowledge base complementing the framework with observations of actor-attributed adversary TTPs from vendor reports.

The framework categorizes the attack life cycle according to different stages, called tactics. Tactics describe the objectives of an adversary within the life cycle of an attack. The tactics within the current framework version are as follows:

1. *Initial Access*: establishing an initial foothold on the target host.
2. *Execution*: how the malicious code is executed on the target host.
3. *Persistence*: all methods to maintain access to the compromised host.
4. *Privilege Escalation*: elevating access to other host or network resources.
5. *Defense Evasion*: all techniques used to avoid detection or other defenses.
6. *Credential Access*: obtaining credentials to expand control of resources.
7. *Discovery*: obtaining contextual awareness of the target system and network.
8. *Lateral Movement*: techniques implemented to pivot across other systems.
9. *Collection*: how information that will be sent to the attacker is collected.
10. *Exfiltration*: transferring information acquired in the process to the attacker.
11. *Command and Control*: the attacker exerting control over the infected host(s).

Each tactic serves as a class of the techniques implemented to accomplish that tactic. For example, to establish persistence (tactic), malware can add a run key to the registry (technique). The framework's techniques describe the steps taken on a technical level. Refer to Fig. 1 for an overview of the categorization of tactics and relevant techniques within the framework. Section 5 will further elaborate on the results displayed in this figure.

Techniques can either be defined generally or specific and be platform-agnostic or platform-specific, depending on how a technique is implemented.

Process Injection is an example of a general technique that applies to several platforms in many variations, where Regsvr32 is a technique with a very specific use case only applicable to the Windows platform. Techniques might appear within two or three tactic categories if applicable. An example of such technique can be seen in Fig. 1. The Scheduled Task technique appears under the Execution, Persistence and Privilege Escalation tactic, as the same technique can be employed within three tactical contexts. This paper will further reference the Windows techniques from the Enterprise ATT&CK framework, as our research focuses on observations of post-compromise activity in Windows malware.

4 Methodology

Dataset. The malware samples used for this research were gathered from Malpedia, a collaborative research platform with an established corpus of malware maintained by Fraunhofer FKIE [21]. The maintainers adhere to the requirements for correctly composed malware datasets, as described by Rossow et al. [23]. This implies that, among other requirements, the maintainers attempt to balance their dataset to avoid it being overshadowed by polymorphic families and that malware samples are annotated with family names and further metadata. Furthermore a carefully curated dataset should favor quality over quantity in order to yield representative results [1,2]. Malpedia aims to provide researchers with a regulated corpus representative of prevalent and timely malware families and code evolution within the malware ecosystem [21]. As a result a curated dataset like Malpedia provides stronger quality assurances than a corpus of malware gathered in the wild.

This research concentrates on Windows malware. Most malware is Windows malware, as Windows is the predominant end-user platform for enterprise and private use. At the time we collected our sample, the corpus of Windows malware available from Malpedia contained 951 unique families first observed in 2007 until 2018. From each Windows malware family collected from Malpedia we have selected the most recent sample based on data from VirusTotal to increase the probability of observing C&C traffic during runtime. For malware families consisting of multiple modules, such as stagers, we have selected the most recent version of each module. The malware family names mentioned within this paper are derived from Malpedia's metadata.

Experimental Environment. For the analysis of the samples in the malware corpus we have used Joe Sandbox Cloud, a public environment for malware analysis [9], formerly known as JoeBox [6]. It combines automated dynamic analysis with basic features for static and network analysis. Given that Malpedia does not include malware that strictly requires a version of Windows later than Windows 7 [21], we have deployed the samples in a sandbox running Windows 7 32-bit or 64-bit depending on detected target architecture, with a configuration of commonly used applications. The reports generated by the sandbox reference generic system calls that can be replicated using other analysis environments.

ATT&CK Mapping. In order to map malware activity reported by the sandbox engine to techniques described within the ATT&CK framework we have built a reference file. Through this the behavior signatures generated by the sandbox during the analysis of the malware corpus were mapped to their corresponding framework technique. The rationalization to plot a particular sandbox signature to an ATT&CK technique is justified by the definition of that technique in the ATT&CK knowledge base [29]. As an example, the sandbox detection signature for the execution of the *IsDebuggerPresent* function was mapped to ATT&CK technique 1063, Security Software Discovery, as checking for active debuggers is implemented by malware authors as a form anti-debugging. As a sandbox can detect different behaviors which all map to one ATT&CK technique, one technique can be described by multiple behavior signatures. Within our results, different variations of one ATT&CK technique within an individual malware family only count once towards that technique. Not every activity logged by the sandbox is necessarily malicious. If behavior described in a sandbox signature did not accurately match a framework technique, it was not mapped. Of 861 unique behavior signatures outputted by the sandbox, 398 were mapped to an ATT&CK technique.

5 ATT&CK Techniques in Windows Malware

This section discusses the results of the analysis of the malware corpus and their mapping onto the ATT&CK framework. Figure 1 shows a heatmap of the ATT&CK matrix, with color shading representing the number of observations of each technique within our analysis. Table 1 shows an overview of the ATT&CK techniques most common in our analysis. As one sample can perform multiple executions of the same technique during runtime (e.g. inject into multiple processes), this is counted as one instance of that technique towards our results.

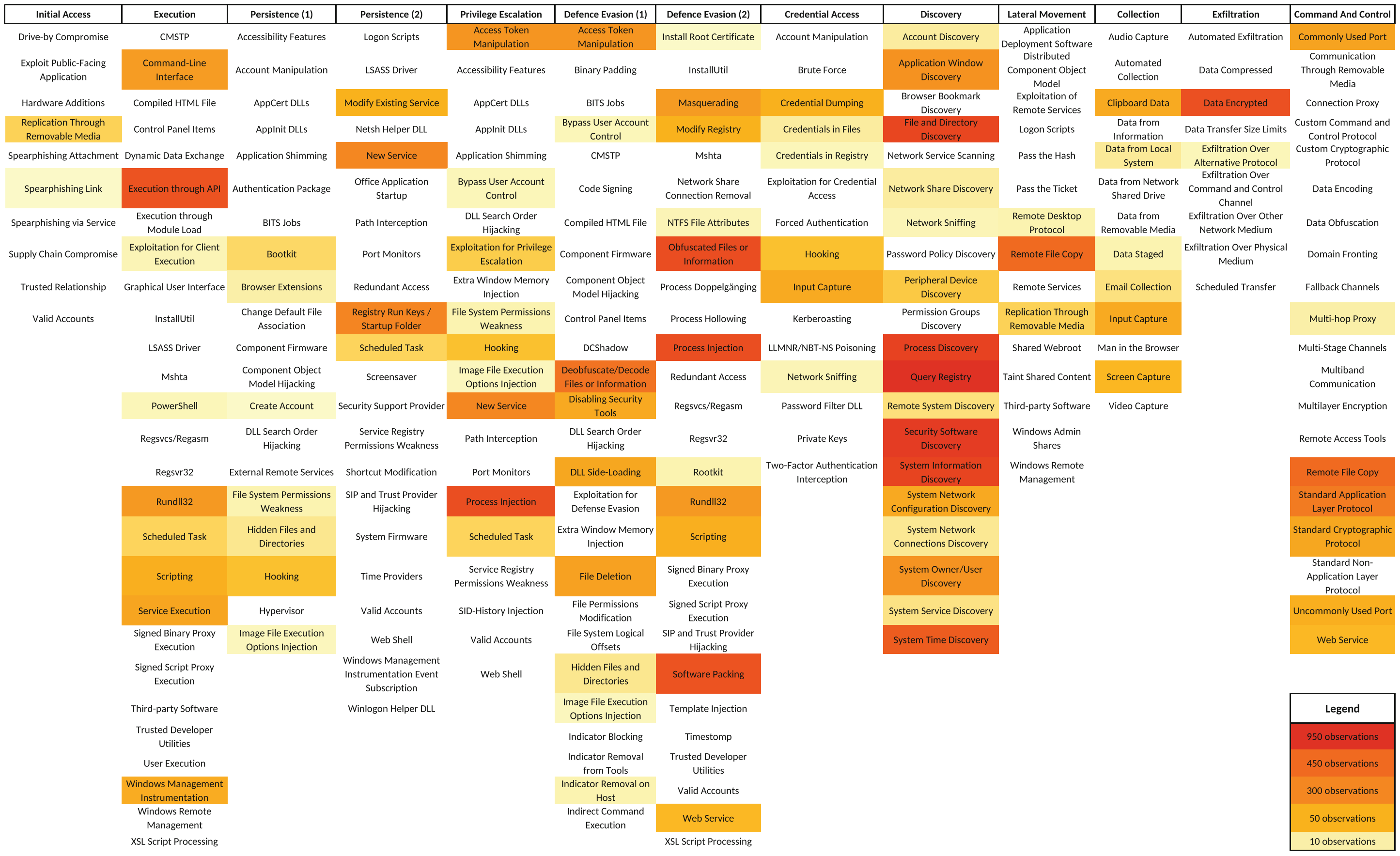
The total of techniques observed in our analysis gravitates towards the inner tactics of the framework, as observed from Fig. 1. The locus of activity in intermediate stages is inherent to the nature of automated malware analysis, which focuses on host and network artifacts from malware runtime. In Sect. 7, we will focus on the limitations of automated analysis and implications for CTI.

Below we will discuss the observations from our analysis based on the categorization offered by the ATT&CK framework. Each technique is discussed within the context of its tactic. Techniques that apply to multiple tactics will be discussed in the context of the tactic justified by our analysis results.

5.1 Execution

The Execution tactic comprises all techniques through which a malicious actor can execute code on a target system. Below the most observed techniques for this tactic from the framework are discussed.

Execution through API. We have observed 562 malware samples launching processes by calling the *CreateProcessA* function Windows application programming interface (API). Furthermore we have observed many instances of dynamic



Initial Access	Execution	Persistence (1)	Persistence (2)	Privilege Escalation	Defence Evasion (1)	Defence Evasion (2)	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
Drive-by Compromise	CMSTP	Accessibility Features	Logon Scripts	Access Token Manipulation	Access Token Manipulation	Install Root Certificate	Account Manipulation	Account Discovery	Application Deployment Software	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	Command-Line Interface	Account Manipulation	LSASS Driver	Accessibility Features	Binary Padding	InstallUtil	Brute Force	Application Window Discovery	Distributed Component Object Model	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Compiled HTML File	AppCert DLLs	Modify Existing Service	AppCert DLLs	BITS Jobs	Masquerading	Credential Dumping	Browser Bookmark Discovery	Exploitation of Remote Services	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Netsh Helper DLL	AppInit DLLs	Bypass User Account Control	Modify Registry	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Information	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	New Service	Application Shimming	CMSTP	Mshta	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through API	Authentication Package	Office Application Startup	Bypass User Account Control	Code Signing	Network Share Connection Removal	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through Module Load	BITS Jobs	Path Interception	DLL Search Order Hijacking	Compiled HTML File	NTFS File Attributes	Forced Authentication	Network Sniffing	Remote Desktop Protocol	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Port Monitors	Exploitation for Privilege Escalation	Component Firmware	Obfuscated Files or Information	Hooking	Password Policy Discovery	Remote File Copy	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Redundant Access	Extra Window Memory Injection	Component Object Model Hijacking	Process Doppelgänging	Input Capture	Peripheral Device Discovery	Remote Services	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallUtil	Change Default File Association	Registry Run Keys / Startup Folder	File System Permissions Weakness	Control Panel Items	Process Hollowing	Kerberoasting	Permission Groups Discovery	Replication Through Removable Media	Input Capture		Multi-hop Proxy
	LSASS Driver	Component Firmware	Scheduled Task	Hooking	DCShadow	Process Injection	LLMNR/NBT-NS Poisoning	Process Discovery	Shared Webroot	Man in the Browser		Multi-Stage Channels
	Mshta	Component Object Model Hijacking	Screensaver	Image File Execution Options Injection	Deobfuscate/Decode Files or Information	Redundant Access	Network Sniffing	Query Registry	Taint Shared Content	Screen Capture		Multiband Communication
	PowerShell	Create Account	Security Support Provider	New Service	Disabling Security Tools	Regsvcs/Regasm	Password Filter DLL	Remote System Discovery	Third-party Software	Video Capture		Multilayer Encryption
	Regsvcs/Regasm	DLL Search Order Hijacking	Service Registry Permissions Weakness	Path Interception	DLL Search Order Hijacking	Regsvr32	Private Keys	Security Software Discovery	Windows Admin Shares			Remote Access Tools
	Regsvr32	External Remote Services	Shortcut Modification	Port Monitors	DLL Side-Loading	Rootkit	Two-Factor Authentication Interception	System Information Discovery	Windows Remote Management			Remote File Copy
	Rundll32	File System Permissions Weakness	SIP and Trust Provider Hijacking	Process Injection	Exploitation for Defense Evasion	Rundll32		System Network Configuration Discovery				Standard Application Layer Protocol
	Scheduled Task	Hidden Files and Directories	System Firmware	Scheduled Task	Extra Window Memory Injection	Scripting		System Network Connections Discovery				Standard Cryptographic Protocol
	Scripting	Hooking	Time Providers	Service Registry Permissions Weakness	File Deletion	Signed Binary Proxy Execution		System Owner/User Discovery				Standard Non-Application Layer Protocol
	Service Execution	Hypervisor	Valid Accounts	SID-History Injection	File Permissions Modification	Signed Script Proxy Execution		System Service Discovery				Uncommonly Used Port
	Signed Binary Proxy Execution	Image File Execution Options Injection	Web Shell	Valid Accounts	File System Logical Offsets	SIP and Trust Provider Hijacking		System Time Discovery				Web Service
	Signed Script Proxy Execution		Windows Management Instrumentation Event Subscription	Web Shell	Hidden Files and Directories	Software Packing						
	Third-party Software		Winlogon Helper DLL		Image File Execution Options Injection	Template Injection						
	Trusted Developer Utilities				Indicator Blocking	Timestomp						
	User Execution				Indicator Removal from Tools	Trusted Developer Utilities						
	Windows Management Instrumentation				Indicator Removal on Host	Valid Accounts						
	Windows Remote Management				Indirect Command Execution	Web Service						
	XSL Script Processing					XSL Script Processing						

Legend
950 observations
450 observations
300 observations
50 observations
10 observations

Fig. 1. ATT&CK enterprise matrix for Windows heatmap based on observations for each technique.

linking to the Windows API in order to call functions required for the malware to fully execute. Although we have observed a decrease in the implementation of this technique, accounting for 5.88% of total techniques observed in 2009 to 3.09% in 2018, this remains an efficient execution technique as host-based mitigation of specific API calls leads to undesirable side-effects.

Rundll32. We have seen 175 samples of malware being capable of executing a dynamic-link library (DLL) via *rundll32.exe*. This technique is deployed by malware for execution as well as defense evasion. Furthermore it can be used several times to launch additional modules. Using this technique provides an attack vector difficult to monitor for as it is also used by benign Windows functions.

Command-Line Interface. Within our dataset, 161 samples interact with the host system via the command-line interface, *cmd.exe* for the execution of modules. Several trojan families identified by Malpedia as EvilBunny, Oceansalt, Remcos, Sword and WebC2 invoke *cmd.exe* to setup a backdoor by creating a TCP reverse shell. Starting 2017, we have observed an increase in the use of obfuscated command line arguments with *cmd.exe*, apparently to evade signature-based detection measures.

Service Execution. Another prominent execution vector found is to register or execute as a service. Found in 115 samples, the observed implementation of this technique seems to have decreased, accounting for 8.33% of total techniques observed in 2012 to 1.07% in 2018. A handful of malware families in our dataset (Carbanak, Koadic, OlympicDestroyer, NetC) has been observed being capable of executing remote processes via *PsExec.exe*. As Service Execution directly executes the service, it is different from the New Service technique, which is used as a persistence tactic and described in the Persistence section.

Within our dataset we have observed the recent emergence of malware that only exists in memory, known as fileless malware. Not a novel finding in itself, we found that the emergence of fileless malware in our dataset overlaps with the first coverage of the topic in scientific literature [16]. Below we will shortly focus on malware employing PowerShell and WMI for fileless execution.

PowerShell. Within our dataset, 7 families used the PowerShell command-line for execution. All samples were first observed by VirusTotal in either 2017 or 2018, from families identified by Malpedia as Emotet, Rozena, DNSMessenger, Ramnit, DownPaper, SnatchLoader and Empire Downloader. Emotet, Ramnit and SnatchLoader executed PowerShell and called *CreateObject* to create a shell object to download and subsequently execute second-stage malware. Rozena was observed to attempt to create a reverse shell using several encrypted shell scripts called upon through PowerShell.

Windows Management Instrumentation. We found 82 samples accessing WMI, for example extracting information about the operating system or installed anti-virus software. As a subset we observed 7 families using the WMI command-line (WMIC) in the execution of malicious code. We have identified these samples belonging to families identified on Malpedia as Moker, EternalPetya, Spora,

Table 1. Number of observations per ATT&CK technique in our dataset.

ATT&CK technique	Count	ATT&CK technique	Count
Query Registry	950	Windows Management Instrumentation	82
Security Software Discovery	748	Scripting	71
Process Discovery	684	Uncommonly Used Port	67
System Information Discovery	669	Credential Dumping	56
File and Directory Discovery	658	Modify Existing Service	53
Obfuscated Files or Information	604	Modify Registry	50
Process Injection	597	Screen Capture	48
Data Encrypted	576	Web Service	47
Execution through API	562	Hooking	41
Software Packing	558	Peripheral Device Discovery	35
System Time Discovery	506	Exploitation for Privilege Escalation	33
Remote File Copy	423	Replication Through Removable Media	30
Deobfuscate/Decode Files or Information	378	Scheduled Task	29
Standard Application Layer Protocol	338	Bootkit	26
Registry Run Keys / Start Folder	287	Remote System Discovery	21
New Service	273	Email Collection	20
Application Window Discovery	216	System Service Discovery	19
System Owner/User Discovery	210	Hidden Files and Directories	16
Access Token Manipulation	197	System Network Connections Discovery	15
Rundll32	175	Data from Local System	14
Masquerading	165	Credentials in Files	12
Command-Line Interface	161	Account Discovery	12
File Deletion	135	Network Share Discovery	12
Commonly Used Port	129	Browser Extensions	10
Service Execution	115	Multi-hop Proxy	10
DLL Side-Loading	106	File System Permissions Weakness	8
Standard Cryptographic Protocol	104	Rootkit	8
Disabling Security Tools	98	Indicator Removal on Host	8
System Network Configuration Discovery	97	Data Staged	8
Clipboard Data	94	Remote Desktop Protocol	8
Input Capture	94	NTFS File Attributes	7

LatentBot, ISFB, Dropshot, EvilBunny, Ghost RAT, Betabot. All of the samples which created processes via WMIC were first observed in 2017, except for Moker in 2015. Out of these families, 5 are attributed to a sophisticated actor group (EternalPetya, ISFB, Dropshot, EvilBunny, Ghost RAT).

From the deployment of techniques for Execution, we can observe a few trends. Execution through API and Service Execution seem to have decreased and the use of obfuscated command lines increased. We have also identified increasing proliferation of WMI and PowerShell for fileless execution, in order to circumvent common preventive controls such as application whitelisting tools and leaving no on-disc forensic evidence. With the observed increase of obfuscated *cmd.exe* command-lines and fileless execution vectors, this indicates attackers are innovating their execution techniques to establish a foothold on target hosts. This also shows how the ATT&CK framework is useful to identify trends in technique adoption within a tactic deployed by attackers.

5.2 Persistence

In order to endure presence on the target system, malware authors employ various techniques. Below we review the most observed techniques for this ATT&CK tactic, including two techniques which have increased over recent years.

Registry Run Keys/Start Folder. Adding an autostart key to either the Windows registry or startup folder is an established persistence technique amongst malware authors, observed in 287 samples from the dataset. Dropping portable executable (PE) files to the startup folder directly is another variation of this technique, seen in 28 samples.

New Service. We have observed 273 samples being capable of creating a new service to be executed at Windows startup. Using the *CreateServiceA* function and adding malicious DLLs are popular methods observed in various types of malware over time, both from sophisticated as lower-level malware authors.

Modify Existing Service. We have observed 53 families implementing persistence by adjusting services, either by modifying registry keys using *reg.exe* in *HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\* or using *sc.exe* to modify the status Windows services, such as Windows Update.

Hooking. We found 41 samples capable of hooking various software functions. Particular examples being banking and POS malware families hooking browser-specific functions. The deployment of hooking techniques is relatively stable over the timeframe analyzed. We have observed hooking deployed in more sophisticated families such as Snifula, Babar, EquationGroup, Nymaim, DanaBot and QakBot. Having observed this technique in sophisticated families and being difficult to mitigate because it abuses fundamental features of the operating system, this is an important attack vector to monitor for.

Scheduled Task. We have observed the implementation of task scheduling to have increased from 2015, using *at.exe* and *schtasks.exe* to trigger execution on every reboot or even every minute. It is mostly recent ransomware from 2017 and 2018 employing this technique, such as CryptoWire, Jaff, Rapid Ransom and Sage.

Image File Exection Options Injection. Four malware families from 2017 and 2018 have been observed to perform Image File Execution Options Injection in order to launch a new process by attaching a debugger to a current process.

The implementation of the most observed persistence techniques is relatively stable. The usage of Registry Run Keys / Start Folder is a known common technique, but the increase in task scheduling and recent observations of Image File Execution Options Injection indicates that attackers are seeking new techniques to bypass common preventive controls in maintaining access to an infected host.

5.3 Privilege Escalation

The techniques described within this tactic are implemented to establish a higher level of permissions to further increase control over the infected host or network. Below we cover three techniques most seen within this tactic.

Process Injection. Of the methods to launch malicious code, process injection is the most popular execution technique in our results, found in 597 of 951 families. The ATT&CK framework recognizes several subtypes of process injection for Windows, all of which are observed within our dataset. Of the defined sub-techniques, we have observed 86 instances of DLL injection and 153 samples being capable of portable executable injection. Furthermore we have found thread execution hijacking in 101 samples and thread local storage (TLS) callback injection in 51 samples. We have found asynchronous procedure call (APC) injection in a total of 37 samples. Relative to other methods, PE injection is the most implemented process injection technique in our dataset.

Access Token Manipulation. Being used in 197 samples from our dataset, this method to manipulate the ownership of active Windows processes is also a popular technique to escalate privileges. The most common implementation of this technique found in our dataset is through subsequent calls to the *OpenProcessToken*, *LookupPrivilegeValueA*, *AdjustTokenPrivileges* functions.

Exploitation for Privilege Escalation. We have observed 33 samples attempting to access the *ShellExecute* function, for which a privilege escalation vulnerability was published in 2014 [17].

Some techniques described in previous tactics can be employed to concurrently escalate privileges. A Scheduled Task for execution can also elevate privileges to SYSTEM. Creating a New Service within a persistence tactic can launch a service with administrator privileges to execute under escalated SYSTEM privileges. Although not much evolution between the different approaches to process injection was observed, it remains the primary vector for elevating privileges. By distinguishing multiple approaches to this technique, the ATT&CK framework facilitates a more informed discussion on the mitigation of such attacks.

5.4 Defense Evasion

Malware authors deploy several techniques in a tactic to avoid or subvert detection or mitigation technologies. We have observed four commonly deployed techniques, with DLL side-loading being on the rise.

Obfuscated Files or Information. This ATT&CK technique serves as a holder of all methods to draw malicious artifacts difficult to detect by obfuscating its contents in transit or at rest. Our dynamic analysis has found 593 malware samples with obfuscated instructions. We have also observed 37 malware samples with .NET source code containing either long sections of Base64 encoded code, as well the .NET code calling decryption functions *CreateDecryptor*. Furthermore we have found 40 samples of malware with inlined NOP slides, which suggests the presence of obfuscated (shell) code.

Software Packing. Packing has become a standard measure to make malicious files more difficult to detect or analyze. Based on *zlib* compression ratios, our sandbox detected a total 558 malware samples employing some form of packing. With regards to specific packers, UPX is a commonly used packer observed in 54 samples. We have observed 15 samples using RAR archiving for packing.

Table 2. Relative implementation of DLL side-loading from 2011 to 2018.

year	2011	2012	2013	2014	2015	2016	2017	2018
% of total	0.36%	0.29%	0.22%	0.18%	0.51%	0.63%	0.69%	0.82%

Deobfuscate/Decode Files or Information. We have observed 359 samples of malware using string decryption functions to recover obfuscated code sections. Like packing, this technique is deployed to hide malicious code in order to make it more difficult to detect. Encoding only the malicious sections of a malicious file and decoding them before execution might evade heuristic detection of malware.

Masquerading. All methods to manipulate or abuse names and locations of legitimate files to evade defenses are grouped under this technique. We have observed 165 instances of masquerading within our dataset, such as creating a presence in the Program Files, Windows and driver directories. Furthermore we found 101 files creating files within the *system32* directory. 31 samples were observed creating executable files named similar to existing Windows files, 19 other samples did employ names of commonly used third-party applications.

DLL Side-Loading. We have observed 106 unique instances of DLL side-loading, with 90 first observed from 2016 to 2018. Increasing from 0.36% of total techniques detected in 2011 to 0.83% in 2018 as seen in Table 2, we expect this technique to keep increasing.

Packing and obfuscating sections of malicious code are standard measures for malware authors with the above techniques being commonly observed. The recent rise of DLL side-loading suggests that attackers are innovating their techniques in order to ensure evasion of established mitigations.

5.5 Credential Access

This tactic describes techniques to obtain some form of privileged credentials to be used in later stages of an attack. Below, Input Capture and Credential Dumping will be discussed, which are the most prevalent techniques observed.

Input Capture. We have found 94 samples capable of capturing user input. 54 samples did implement a global keyboard hook with the *SetWindowsHookEx* function to intercept keystrokes. 31 samples were observed implementing functionality to retrieve information about pressed keystrokes using functions such as *GetAsyncKeyState*, *GetKeyState* and *MapVirtualKeyA*. 8 samples created a DirectInput object using the *DirectDrawCreateEx* function to capture keystrokes.

Credential Dumping. This technique describes all means to obtain login and password information from the operating system and software, which may later be used for lateral movement on the network. We have found 56 samples implementing techniques as harvesting browser history and passwords (44 samples) and 9 samples querying for file locations and registry keys of common third-party FTP tools. Another 9 samples queried the *Login Data* registry key used by Chrome and *IntelliForms2*, used by Internet Explorer to store passwords.

We have also observed 12 samples querying the file system and 6 samples searching the Windows registry for stored credentials. Hooking, already discussed within the context of Persistence, is also a technique for Credential Access. Though we have less results for this tactic, from our dataset it can be observed that keylogging is the main method to capture user input to obtain credentials, followed by dumping credentials from installed software.

5.6 Discovery

The Discovery tactic provides the basis for success of later attack stages. It is essentially a second iteration of reconnaissance, consisting of techniques an attacker deploys to gather information about an infected system and its placement in the network. Many of the most observed techniques within our analysis are part of the Discovery tactic. As most Discovery techniques deploy native operating system functions, this activity is well-detected by dynamic analysis environments, but also difficult to detect against.

Query Registry. Querying the Windows registry to discover information about the host system is the most common technique within our dataset, seen in 950 samples. Most of the samples within our dataset have a capability to read software restriction policies from the Windows registry by enumerating *HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers*. The DWORD value of AuthenticodeEnabled indicates whether the execution of binaries is restricted by the OS. 345 samples have the ability to query the machine globally unique identifier (GUID) from the registry, most presumably as a unique identifier of the infected system.

Security Software Discovery. We have found 748 samples capable of detecting the presence of security features such as anti-virus software, local firewall rules and virtualization software. We also observed a significant increase in the implementation of security software discovery from 2010 to 2018, of which Table 3 provides an overview. Anti-debugging is the most detected specific implementation of this technique in 519 samples, by querying the *SystemKernelDebuggerInformation* function to detect a ring 0 debugger being attached to the current process. 385 samples were observed detecting a debugger by checking the time difference between two Windows API calls, *GetProcessHeap* and *CloseHandle*. 349 performed an API call to *IsDebuggerPresent*. Checking the presence of a debugger by setting *GetLastError* in the registry to a random value and checking whether it has changed after calling *OutputDebugString* is observed in 92 samples. 70 samples have been observed executing a Read-Time-Stamp-Counter (RDTSC) instruction to determine the speed with which instructions are executed by the processor, of which the presence of a debugger might be inferred. This specific method, first observed in 2012, has gained traction with 52 out of 70 samples first observed in 2016 or later.

Table 3. Relative implementation of security software discovery from 2010 to 2018.

year	2010	2011	2012	2013	2014	2015	2016	2017	2018
% of total	6.13%	8.73%	8.47%	9.38%	11.55%	12.46%	12.32%	11.80%	11.68%

The detection of virtual machines or sandbox environments is another popular form of anti-analysis that has gained adoption over the last couple of years. We have observed 187 samples being able to detect various virtualization products by detecting registry keys specific to guest sharing functionality, such as *HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VMTools*. 44 samples were observed calling the *PhysicalDrive0* function to check for strings that might indicate the drive being virtualized.

Process Discovery. Within the ATT&CK framework Process Discovery describes all techniques to gather information about active processes on the host system. Within our dataset 599 samples implement this technique during runtime through calls to Windows functions such as *CreateToolhelp32Snapshot*, *Process32First*, *Process32Next*. 7 samples first observed from 2016 use *tasklist.exe* to discover running processes on both local and remote systems.

System Information Discovery. This technique, supporting further execution of the malware by querying operating system and hardware artifacts, is implemented by 669 of the samples in our dataset. The most common implementation is querying the Windows version using the *GetVersion* function, observed in 399 samples. Retrieving locale information such as the language of the user interface by querying the *GetLocaleInfoA* or *GetLocaleInfoEx* functions of the Windows API is used in 399 samples. 152 samples get this information in a similar way, using *VirtualQuery* and *VirtualAlloc* to gather information about the memory contents. 179 samples have been observed to check CPU instructions, which might have anti-analysis purposes. Depending on the call, the instruction can return the CPU's manufacturer ID string, but also the hypervisor brand. Certain return values might indicate whether the malware is running on a physical or virtual machine. 24 samples retrieved processor information from the Windows registry key *HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor*.

System Network Configuration Discovery. 97 of the samples in our dataset used variations of this technique. Of these samples, 60 called the *GetAdaptersInfo* function to retrieve information about the network adapter. Other samples have been observed using *ipconfig*, *netstat* or *netsh* to lookup Windows network configuration. Within this technique category, we have seen 24 samples querying standard online IP and geolocation services to determine the online IP address of the infected system.

Based on our analysis we see that Discovery is standard practice for malware. For Security Software Discovery, we have observed an increase. Within that technique, we see that the use of an RDTSC instruction to detect debuggers has proliferated since 2016. Most discovery techniques used in malware blend in with the flow of benign applications as they rely on native operating system functions. This makes Discovery also a tactic difficult to mitigate, which makes a case for application whitelisting to prevent the execution of malicious software early in the life cycle.

5.7 Lateral Movement

Lateral Movement describes all techniques implemented to pivot over the network to other systems of interest. The techniques of this tactic are difficult to observe with dynamic analysis as many techniques depend upon manual attacker intervention to pivot over the network, which is why this tactic touches less host artifacts anyway. Also not every malware family might deploy lateral movement techniques. Therefore the observations of techniques from this tactic are limited. We will shortly discuss the techniques found for this tactic.

Remote File Copy. This technique describes malicious download and upload activity within the network, as well to adversary-controlled infrastructure. We have detected 423 instances of Remote File Copy, which mainly consists of 336 samples attempting to download additional files as detected by our analysis environment. 101 samples communicated using plain HTTP GET requests, most of them storing result in the Temporary Internet Files directory. 18 malware families, among which Bagle, Bundestrojaner, Ransomlock, Redalpha and Yty, established HTTPS connections.

Replication Through Removable Media. We have found 30 samples trying to infect USB storage devices by creating *autorun.inf* files with an *Open* or *ShellExecute* entry.

Remote Desktop Protocol. 8 samples were observed trying to start the Remote Desktop service, which can be an effective stealth technique for lateral movement, as it blends in with the normal network protocol flow.

Our observations for this tactic are limited, as lateral movement is generally a non-automatic process, involving manual operations as the attacker pivots over the network. However we found a significant number of 336 samples attempting to download additional files, which makes a case for host monitoring of unusual processes establishing a network connection.

5.8 Collection

The Collection tactic describes techniques deployed to gather sensitive information. We found that the most observed techniques for this tactic all rely on native Windows functions to acquire sensitive user information. This tactic is difficult to prevent or detect, as the attacker did already bypass several defenses and gained considerable foothold.

Clipboard Data. In our analysis 94 malware samples attempted to obtain data from Windows clipboard. In 48 instances, samples performed subsequent calls to *OpenClipboard* and *GetClipboardData*. 21 samples started a window in the clipboard class *CLIPBRDWNDCLASS* to obtain copy-paste operations.

Screen Capture. 48 samples tried to capture GUI contents, primarily with calls to functions such as *GetDesktopWindow*, *GetWindowRect* to retrieve window dimensions and *BitBlt* and *GetDIBits* to store the capture in a buffer.

Email Collection. 20 samples actively collected of email messages by querying file locations and registry keys associated with mail clients such as Windows Mail client and Outlook.

As with lateral movement, we recognize that actual collection is difficult to capture using dynamic analysis, which results in fewer observations within this tactic. We however suspect that the use of scripting, of which we have found 71 instances, to automatically search and copy data depending on certain criteria is also deployed for Collection purposes.

5.9 Exfiltration

The exfiltration tactic describes all techniques implemented to exfiltrate data from the target to the attacker. Reporting of exfiltration depends on observation of actual exfiltration attempts, which are difficult to capture with dynamic analysis. Furthermore not all attackers apply this tactic, as some (e.g. ransomware authors) are not interested in exfiltrating data. We have found 576 being capable of encrypting local data, which is described with the Data Encrypted technique. We expect this number to be distorted by 27 samples of ransomware in our dataset, which encrypts data but not for exfiltration purposes. Furthermore we have found 6 samples capable of uploading files via FTP as these samples called to the *FtpPutFile* function, identified as Exfiltration Over Alternative Protocol.

5.10 Command and Control

Within this tactic, the attacker is accessing the target network from a remote location. In our analysis, 67 samples were observed to establish TCP or UDP traffic on non-standard ports. This technique, described as Uncommonly Used Port within ATT&CK, is known to be deployed to circumvent improper firewall and proxy configurations. We have also found 47 samples being capable of communicating with popular social media such as Facebook, Tumbler and paste sites such as Pastebin, which are frequently used for C&C. Within the ATT&CK framework this use case is classified under the Web Service technique. For the C&C technique Multi-hop Proxy, we have detected 11 samples initiating a Tor connection. Section 6 elaborates on the deployment of named pipes for C&C.

6 Adoption of Sophisticated Techniques

Within our analysis we have observed named pipes being implemented for C&C. Previously exclusively implemented in malware attributed to sophisticated actors, this technique is observed to have been adopted by less sophisticated malware authors. The attribution of malware to actor groups is part of

Table 4. New samples deploying IPC named pipes per year.

year	2010	2011	2012	2013	2014	2015	2016	2017	2018
observations	0	0	1	4	4	4	7	16	18

the malware metadata in Malpedia, the repository from which we have gathered our samples. For its attribution of malware families to actors, Malpedia relies on reporting from security vendors and independent security researchers. For instance, the samples of Pupy available within Malpedia are attributed to Iranian actors based on reports from 6 individual sources [15].

Named pipes are a method for inter-process communication (IPC), both with local and remote processes. Dynamic malware analysis is suited for discovering this technique, as a process is expected to call the *CreateNamedPipe* function of the *kernel32.dll* kernel module to create a named pipe. The named pipe server allows both local and remote processes to connect to the pipe and exchange information with the malware. As it can set up communication via SMB and RPC, is potentially also a technique that can be deployed to evade detection of command and control (C&C) traffic. By setting up one compromised host as internal C&C server to handle outbound traffic and having other compromised hosts connect on a peer-to-peer basis via named pipes, the footprint of network traffic is reduced considerably, which also reduces the odds of detection. The earliest sample of malware known to implement named pipes for communication with remote hosts is a variant of Conficker first observed in 2009 [22]. It also reported to be deployed for C&C by the Duqu family [27].

The ATT&CK technique definition of Process Injection states that, apart from the Windows implementations of the techniques described, 'more sophisticated samples' of malware may use named pipes or other IPC mechanisms as a communication channel [28]. As named pipes may also be connected to remote processes over SMB and RPC, it can also be deployed as a technique for C&C.

The detection of this technique within our dataset is in line with the Mitre statement that named pipes are specific to the more advanced malware families. As shown in Table 4, we have observed an increase in the use of named pipes within more recently observed samples. What attracts attention specifically is that out of the 47 samples where our analysis environment has observed this technique being deployed, 38 samples were first observed in 2017 and 2018. Except for samples from two families (Dorkbot, Snifula), based on Malpedia metadata, all samples prior to 2015 are attributed to sophisticated actor groups. From 2016, the technique is observed in malware attributed to sophisticated actor groups (TurnedUp, Pupy, Mosquito, EternalPetya, PandaBanker, OlympicDestroyer), but also in common crimeware such as Zeus, Karius, Trickbot, AlinaPOS, QakBot and ransomware (Gandcrab, PyLocky). This indicates that techniques previously attributed exclusively to advanced actor groups are getting adopted by other malware authors.

7 Limitations of CTI from Automated Analysis

In the discussion of our results it became clear that detection of malicious artifacts naturally gravitates towards the intermediate tactics of the attack life cycle. In that sense our analysis exposes issues inherent to malware research with automated analysis. Most of these are known in the malware research field, but as ATT&CK is primarily a CTI model, we will use this chapter to evaluate the biases automated analysis might introduce to the CTI field.

Automated malware analysis suites offer a compelling solution to gain a quick and timely overview of individual or bulk malware threats. Automated analysis observes runtime behavior of malware. Attackers however might employ stealth and deception techniques, such as anti-analysis and evasion. This could result in malware not or only partially being detected, as it does not trigger or branches to deception code. It must also be considered that any analysis environment might not be able to fully detect all behavior exhibited by malware. Results invariably depend on the capabilities of the resource used for analysis. Still these are all known limitations to the concept of automated analysis of malware, as it only reports behavior observed during a time-constrained runtime [31]. But as automated malware analysis focuses on host and network artifacts of malware, it is thus biased towards the reporting of those artifacts. These biases become apparent when using a CTI-oriented model such as ATT&CK. As the requirements for CTI tilt toward the latter stages of the framework and proper CTI must never be biased, we might even argue that automated analysis is an unsuited source when taken by itself.

Initial Access tactics, such as the delivery of a malicious email attachment, are not accounted for during automated analysis. This also expresses in the ATT&CK plotting of our results in Fig. 1. Techniques for Lateral Movement, Exfiltration and C&C activity are difficult to record with a sandbox as the actual execution of these tactics depends on certain preconditions, potential manual attacker involvement and the availability of C&C infrastructure. The ATT&CK framework accounts for deployment of evasive routines in malware within the Defense Evasion tactic, which describes techniques used to evade detection or avoid other defenses. Logically sandboxes only detect evasion to a limited extent.

As a consequence, customers of CTI reports based on automated analysis of malware should be aware of the limitations inherent to the mechanism. An accurate CTI product must take into account the full threat context and consider alternative hypotheses. When used as a source of CTI, automated analysis reports should be treated with a different confidence level than results of manual research. Nonetheless it is evident that CTI benefits from a standardized language like ATT&CK, as it fosters effective dissemination and decision-making.

8 Conclusion

Our work is the first to use the ATT&CK framework to present the results of the analysis of techniques observed during execution of a large sample of malware. Having identified established and emerging techniques from the framework, this research is the first that provides an overview of a representative sample of malware using the ATT&CK framework. Through this, we have demonstrated the benefits of using an common taxonomy for the reporting of TTPs. We have shown this improves the actionability and unambiguous communication of CTI from sandbox analysis results.

We have observed differences in the degree of innovation between the different tactics of the ATT&CK framework. For the execution of malicious code, most malware relies on the Windows API. We have identified an increase in the implementation of fileless execution vectors using WMI and PowerShell. Together with the use of obfuscated command lines, this shows how malware authors are innovating their execution tactic to bypass traditional defenses. We observed innovation in obtaining persistence through the use of task scheduling, complementing established persistence techniques like autostart items and creating a new service. Process injection remains the primary technique for privilege escalation. DLL side-loading seems to be on the rise in order to evade established defenses, complementing more established evasive techniques such as obfuscation and packing. For malware accessing user credentials, capturing user input through keylogging is the most common technique within our dataset. We found that discovery of security software has become standard practice for most malware authors. We have shown different implementations observed for this technique, as well that using RDTSC instructions to detect debuggers has proliferated since 2016. The most common technique for lateral movement we have observed malware is Remote File Copy.

We have shown how C&C via IPC named pipes, previously attributed to sophisticated malware authors, is getting adopted by other actor groups. Through this we identified that malware authors are innovating techniques in order to bypass traditional defense mechanisms.

Acknowledgments. The authors would like to thank the maintainers of Malpedia for providing access to their malware repository and Joe Security for provisioning the sandbox infrastructure. The authors would like to thank VirusTotal for providing access to their API. The ATT&CK mapping built for this research has been shared with Joe Security to develop ATT&CK mapping within their product.

References

1. Barabosch, T., Bergmann, N., Dombeck, A.: Quincy: detecting host-based code injection attacks in memory dumps. In: LNCS (2017)
2. Barabosch, T., Eschweiler, S., Gerhards-Padilla, E.: Bee master: detecting host-based code injection attacks. In: LNCS (2014)
3. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A view on current malware behaviors. USENIX Large-scale exploits and emergent threats (2009)
4. Binsalleeh, H., et al.: On the analysis of the Zeus botnet crimeware toolkit. In: 2010 Eighth International Conference on Privacy, Security and Trust (2010)
5. Chen, X., Andersen, J., Morley Mao, Z., Bailey, M., Nazario, J.: Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In: International Conference on Dependable Systems and Networks (2008)

6. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. **44**, 1–49 (2012)
7. Grill, B., Bacs, A., Platzer, C., Bos, H.: "Nice boots!"-A large-scale analysis of bootkits and new ways to stop them. In: LNCS (2015)
8. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In: International Conference on Information Warfare & Security (2011)
9. Joe Security LLC: Joe Sandbox Cloud Community Edition
10. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L.: Cutting the gordian knot: a look under the hood of ransomware attacks. In: LNCS (2015)
11. Kirat, D., Vigna, G., Kruegel, C.: BareCloud: bare-metal analysis-based evasive malware detection. In: 23rd USENIX Security Symposium (2014)
12. Kirillov, I.A., Beck, D.A., Chase, M.P., Martin, R.A.: The Concepts of the Malware Attribute Enumeration and Characterization (MAEC) Effort (2009)
13. Laliberte, M.: A Twist On The Cyber Kill Chain: Defending Against A JavaScript Malware Attack (2016)
14. Lyda, R., Hamrock, J.: Using entropy analysis to find encrypted and packed malware (2007)
15. Malpedia: win.pupy. malpedia.caad.fkie.fraunhofer.de/details/win.pupy
16. Mansfield-Devine, S.: Fileless attacks: compromising targets without malware. Netw. Secur. **2017**, 7–11 (2017)
17. Microsoft: Microsoft Security Bulletin MS14-027 (2014)
18. Nachreiner, C.: Kill Chain 3.0: Update the cyber kill chain for better defense (2015)
19. Obrst, L., Chase, P., Markeloff, R.: Developing an ontology of the cyber security domain. In: Semantic Technologies for Intelligence, Defense, and Security (2012)
20. O'Kane, P., Sezer, S., McLaughlin, K.: Obfuscation: the hidden malware. IEEE Secur. Privacy **9**, 41–47 (2011)
21. Plohmann, D., Clauss, M., Enders, S., Padilla, E.: Malpedia: a collaborative effort to inventorize the malware landscape. J. Cybercrime & Dig. Investigations, 3 (2018)
22. Porras, P., Saidi, H., Yegneswaran, V.: An analysis of conficker's logic and rendezvous points. Technical Report, Computer Science Laboratory, SRI International (2009)
23. Rossow, C., et al.: Prudent practices for designing malware experiments: status quo and outlook. In: IEEE Symposium on Security and Privacy (2012)
24. Song, D., et al.: BitBlaze: a new approach to computer security via binary analysis. In: LNCS (2008)
25. Sood, A.K., Enbody, R.J.: Crimeware-as-a-service-a survey of commoditized crimeware in the underground market. Int. J. Crit. Infrastruct. Prot. **6**, 28–38 (2013)
26. Strom, B.E., Applebaum, A., Miller, D.P., Nickels, K.C., Pennington, A.G., Thomas, C.B.: MITRE ATT&CK: Design and Philosophy. The Mitre Corporation, McLean, VA, Technical report (2018)
27. Symantec Security Response: W32.Duqu: the precursor to the next Stuxnet. Symantec Security Response (2011)
28. The Mitre Corporation: ATT&CK JSON Library (2018)
29. The Mitre Corporation: Enterprise Matrix - Windows (2018). https://attack.mitre.org/matrices/enterprise/windows/
30. Verizon: 2018 Data Breach Investigations Report. Technical report, New York, NY (2018)
31. Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using CWSandbox (2007)